NestJS Cheat Sheet

Modules · DI · Guards · Pipes · Decorators

SETUP & CLI

```
npm i -g @nestjs/cli
nest new my-app
nest generate module users
nest generate controller users
nest generate service users
npm run start:dev  (watch mode)
```

MODULES

```
@Module({
imports:
[TypeOrmModule.forFeature([User])],
controllers: [UsersController],
providers: [UsersService],
exports: [UsersService],
})
export class UsersModule {}
```

Feature modules keep concerns isolated

CONTROLLERS

```
@Controller('users')
export class UsersController {
constructor(private readonly svc:
UsersService) {}
@Get() findAll(): Promise<User[]> {
return this.svc.findAll() }
@Get(':id') findOne(@Param('id') id:
string) { ... }
@Post() @HttpCode(201) create(@Body()
dto: CreateUserDto) { ... }
@Put(':id') update(@Param('id') id,
@Body() dto) { ... }
@Delete(':id') remove(@Param('id') id) {
... }
```

PROVIDERS & SERVICES

```
@Injectable()
export class UsersService {
constructor(
@InjectRepository(User)
private readonly repo: Repository<User>
) {}
async findAll(): Promise<User[]> {
return this.repo.find()
}
}
```

DTOs & VALIDATION

```
import { IsEmail, IsString, MinLength }
from 'class-validator'
export class CreateUserDto {
@IsString() @MinLength(2)
name: string
@IsEmail()
email: string
}
// main.ts:
app.useGlobalPipes(new ValidationPipe({
whitelist: true }))
```

GUARDS

```
@Injectable()
export class JwtAuthGuard extends
AuthGuard('jwt') {}
@UseGuards(JwtAuthGuard)
@Get('profile')
getProfile(@Request() req) { return
req.user }
@Injectable()
export class RolesGuard implements
CanActivate {
canActivate(ctx: ExecutionContext):
boolean { ... }
}
```

INTERCEPTORS

```
@Injectable()
export class LoggingInterceptor
implements NestInterceptor {
intercept(ctx: ExecutionContext, next:
CallHandler): Observable<any> {
console.log('Before...')
return next.handle().pipe(
tap(() => console.log('After...'))
)
}
}
@UseInterceptors(LoggingInterceptor)
```

MIDDLEWARE

```
@Injectable()
export class LoggerMiddleware implements
NestMiddleware {
use(req: Request, res: Response, next:
Function) {
console.log(`${req.method} ${req.url}`)
next()
}
}
// In module:
configure(consumer: MiddlewareConsumer)
{
consumer.apply(LoggerMiddleware).forRoutes('*')
}
```

EXCEPTION FILTERS

```
@Catch(HttpException)
export class HttpExceptionFilter
implements ExceptionFilter {
catch(ex: HttpException, host:
ArgumentsHost) {
const ctx = host.switchToHttp()
const res = ctx.getResponse<Response>()
res.status(ex.getStatus()).json({
statusCode: ex.getStatus(),
message: ex.message
})
}
}
```

COMMON DECORATORS

```
@Controller @Get @Post @Put @Delete
@Patch
@Param @Body @Query @Headers @Request
@Response
@Injectable @Module @Guard @UseGuards
@UseInterceptors
@Catch @UseFilters  @UsePipes
@SetMetadata('roles', ['admin'])
@Reflector +
Reflector.getAllAndOverride()
```